APPENDIX B

```
/*************************************************************/
/*  Distributed Hypermedia Object Embedding (DHOE) sample    */
/*************************************************************/
...
#include "protocol_lib.h" /* you can find this in the same
                             dirs where libprotocol.a's are */

...

/* X-way to define resources and parse the cmdline args */ typedef
struct{
        int     win;
        int     pixmap;
        int     pixmap_width;
        int     pixmap_height;
        char    *datafile;
} ApplicationData, *ApplicationDataPtr;

static XtResource myResources[] = {
        {"win", "Win", XtRInt, sizeof(int),
         XtOffset(ApplicationDataPtr, win), XtRImmediate, 0},
        {"pixmap", "Pixmap", XtRInt, sizeof(int),
         XtOffset(ApplicationDataPtr, pixmap), XtRImmediate, 0},
        {"pixmap_width", "Pixmap_width", XtRInt, sizeof(int),
         XtOffset(ApplicationDataPtr, pixmap_width), XtRImmediate,
400},
        {"pixmap_height", "Pixmap_height", XtRInt, sizeof(int),
         XtOffset(ApplicationDataPtr, pixmap_height), XtRImmediate,
400},
        {"datafile", "Datafile", XtRString, sizeof(char*),
         XtOffset(ApplicationDataPtr,   datafile),   XtRImmediate,
NULL},
};

static XrmOptionDescRec myOptions[] = {
        {"-win", "*win", XrmoptionSepArg, 0},
        {"-pixmap", "*pixmap", XrmoptionSepArg, 0},
        {"-pixmap_width", "*pixmap_width", XrmoptionSepArg, 0},
        {"-pixmap_height", "*pixmap_height", XrmoptionSepArg, 0},
        {"-datafile", "*datafile", XrmoptionSepArg, NULL},
};

ApplicationData myAppData;

void myDraw()
{
    /* do your drawing... */
    ...

    /* if you draw into your own drawables (myPixmap in this case)
*/
    if (myAppData.win) {
        /* copy from myPixmap to the "shared" pixmap */
        XCopyArea(display, myPixmap, myAppData.pixmap, myGC, 0, 0,
```

```
WIN_WIDTH,
WIN_HEIGHT, 0, 0);
        /* tell Mosaic to update the drawing window */
        send_client_msg(XtNrefreshNotify, display, myAppData.win);
    }
}

void myQuit()

{
        /* tell Mosaic you are exiting... */
        if (myAppData.win)
                send_client_msg(XtNpanelExitNotify,    display,
myAppData.win);

        /* Motif way of exiting */
        XtCloseDisplay(XtDisplay(any widget));

        exit(1);
}

...

main()
{
    Widget app_shell;
    ...

    /* XtInitialize does XOpenDisplay, as well as creates a
toplevel widget */
    app_shell    =    XtInitialize("wt",    "Wt",    myOptions,
XtNumber(myOptions), &argc, argv);
    ...

    /* This func fill up myAppData with the user specified
values/default values */
    XtGetApplicationResources(app_shell, &myAppData, myResources,
XtNumber(myResources), NULL, 0);
    ...

    /* if we have an external window to display the image... */
    if (myAppData.win) {
        XtAddEventHandler(app_shell,    NoEventMask,    True,
handle_client_msg,
NULL);
        register_client(app_shell, display);

        /* register the func to be called when Mosaic exit */
        register_client_msg_callback(XtNexitNotify, myQuit);

        /* tell Mosaic you have started fine */
         send_client_msg(XtNpanelStartNotify,    display,
myAppData.win);
    }
```

```
    ...
    XtMainLoop(); /* Motif's event loop */
}
/*****************************************************************
**********/
```


```
-                                                               -
================================================================
Cheong S. Ang
Software Engineer                email: cheong@ckm.ucsf.edu
UCSF Library & Ctr for Knowledge Mgt    at&t: 415/476-2954
530 Parnassus Avenue, Box 0840           fax: 415/476-4653
San Francisco, California 94143
================================================================
```


```
--PART-BOUNDARY=.19410131723.ZM26058.library.ucsf.edu
Encoding: 21 text
X-Zm-Content-Name: protocol_lib.h
Content-Type: text/plain ; charset=us-ascii

#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>
#include <X11/Xatom.h>

/* for Mosaic to get window id */
#define XtNcommWindow            "Comm Window"

/* msgs from Mosaic */
#define XtNexitNotify            "Exit Notify"
#define XtNmapNotify             "Map Notify"
#define XtNunmapNotify           "Unmap Notify"

/* msgs to Mosaic */
#define  XtNrefreshNotify                "Refresh  Notify"    #define
XtNpanelStartNotify     "Panel Start Notify"
#define XtNpanelExitNotify       "Panel Exit Notify"

extern void handle_client_msg(); extern void register_client();
extern    void    register_client_msg_callback();    extern    void
send_client_msg();

--PART-BOUNDARY=.19410131723.ZM26058.library.ucsf.edu
Encoding: 109 text
X-Zm-Content-Name: protocol_lib.c
Content-Description: plain text
Content-Type: text/plain ; charset=us-ascii

#include "protocol_lib.h"

/* in order to use this API, the program should:

     (1) Call
```

```
        XtAddEventHandler(Widget app_shell,
                    NoEventMask, True, handle_client_msg, NULL);

        (2) Register its widget(window), and the remote display
        register_client(Widget w, Display *remote_display);

        (3) Register the callback functions for each msg
        register_client_msg_callback(char   *msg,   void
    (*function_ptr)());

        (4) The program may also send client_msgs to external
    application by
        send_clinet_msg(char *msg, Display *remote_display, Window
    remote_window);
    */

    typedef void (*FUNPTR)();

    static FUNPTR handle_quit_msg; static Atom EXIT_NOTIFY, MAP_NOTIFY,
    UNMAP_NOTIFY;

    void register_client(w, remote_display)
    Widget w;
    Display *remote_display;
    {
        Window widget_win = XtWindow(w);
        Atom COMM_WINDOW;

            MAP_NOTIFY = XInternAtom(remote_display, XtNmapNotify, 0);
            UNMAP_NOTIFY = XInternAtom(remote_display, XtNunmapNotify,
    0);
            EXIT_NOTIFY  =  XInternAtom(remote_display,  XtNexitNotify,
    0);

            /* for mosaic to get panel window id */
            COMM_WINDOW  =  XInternAtom(remote_display,  XtNcommWindow,
    0);
                XChangeProperty(remote_display,
    DefaultRootWindow(remote_display),
                            COMM_WINDOW, XA_WINDOW,
                            32, PropModeReplace,
                            (unsigned char*)&widget_win, 1);
            XFlush(remote_display);
    }

    void handle_client_msg(w, client_data, event)
    Widget w; caddr_t client_data;
    XEvent *event;
    {
            static int mapped=1;

            if (event->type != ClientMessage) return;
            if (event->xclient.message_type == MAP_NOTIFY) {
                    if (!mapped) {
```

```
                        mapped = 1;
                        XMapWindow(XtDisplay(w), XtWindow(w));
                        XFlush(XtDisplay(w));
                }
        }
        else if (event->xclient.message_type == UNMAP_NOTIFY) {
                if (mapped) {
                        mapped = 0;
                        XIconifyWindow(XtDisplay(w), XtWindow(w),
DefaultScreen(XtDisplay(w)));
                        XFlush(XtDisplay(w));
                }
        }
        else if (event->xclient.message_type == EXIT_NOTIFY) {
                (*handle_quit_msg)();
        }
}

void   register_client_msg_callback(msg_name,   func_ptr)   char
*msg_name;
FUNPTR func_ptr;
{
     if (strcmp(msg_name, XtNexitNotify)==0) {
          handle_quit_msg = func_ptr;
        }
}

void send_client_msg(msg_name, remote_display, remote_window) char
*msg_name;
Display *remote_display;
Window remote_window;
{
     if (strcmp(msg_name, XtNrefreshNotify)==0) {
          XExposeEvent event;

          event.type = Expose;
          event.display = remote_display;
          event.window = remote_window;
          event.send_event = True;
          XSendEvent(remote_display, remote_window, True, Expose,
(XEvent*)&event);
     }
     else {
          XClientMessageEvent event;
          Atom MSG_ATOM;

          MSG_ATOM = XInternAtom(remote_display, msg_name, FALSE);
          event.display = remote_display;
          event.window = remote_window;
          event.type = ClientMessage;
          event.format = 8;
          event.message_type = MSG_ATOM;
          XSendEvent(remote_display,    remote_window,    True,
```

```
NoEventMask, (XEvent*) &event);
     }
          XFlush(remote_display);
}
```